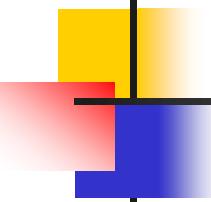




## Chapter 6

# Data Encryption Standard (DES)



# **Chapter 6**

## **Objectives**

- ❑ To review a short history of DES**
- ❑ To define the basic structure of DES**
- ❑ To describe the details of building elements of DES**
- ❑ To describe the round keys generation process**
- ❑ To analyze DES**

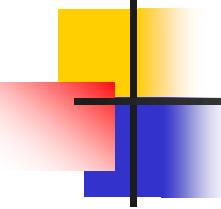
# 6-1 INTRODUCTION

*The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).*

**Topics discussed in this section:**

**6.1.1 History**

**6.1.2 Overview**



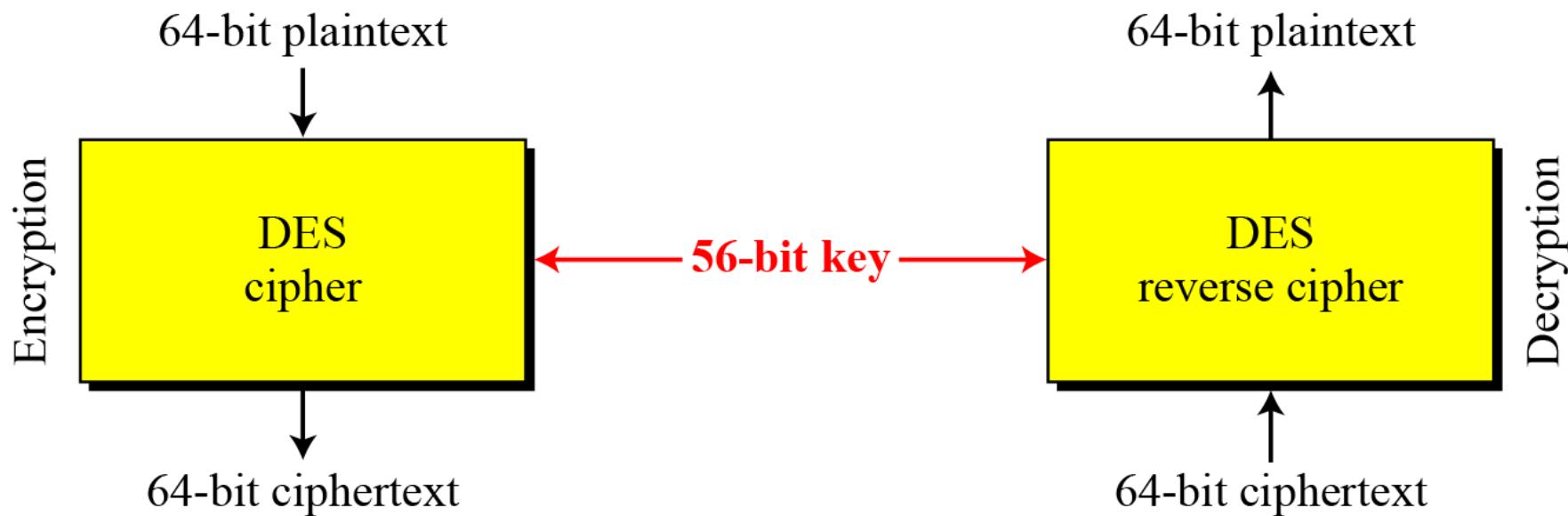
## *6.1.1 History*

*In 1973, NIST published a request for proposals for a national symmetric-key cryptosystem. A proposal from IBM, a modification of a project called Lucifer, was accepted as DES. DES was published in the Federal Register in March 1975 as a draft of the Federal Information Processing Standard (FIPS).*

## 6.1.2 Overview

*DES is a block cipher, as shown in Figure 6.1.*

**Figure 6.1** Encryption and decryption with DES



## 6-2 DES STRUCTURE

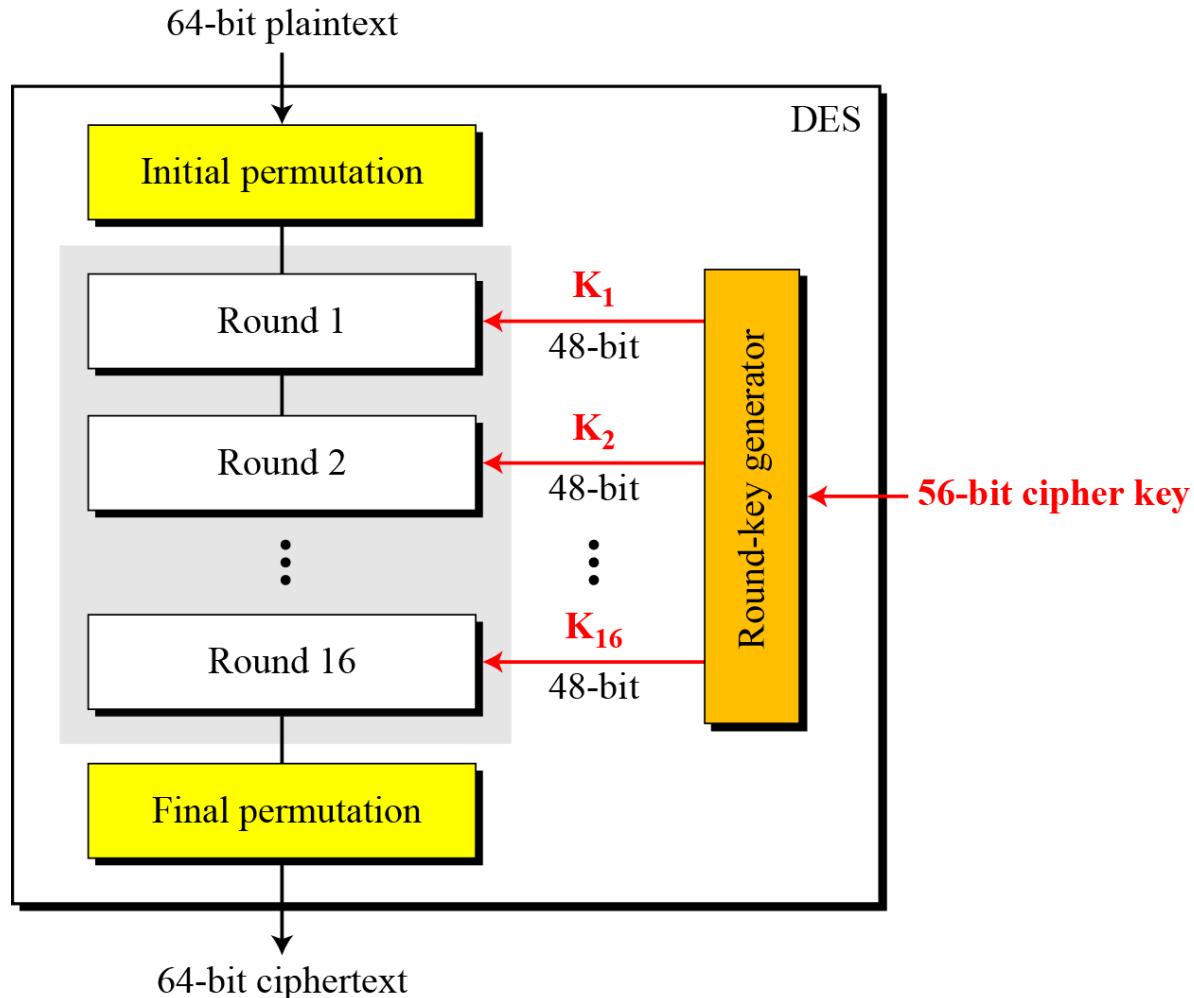
*The encryption process is made of two permutations (P-boxes), which we call initial and final permutations, and sixteen Feistel rounds.*

**Topics discussed in this section:**

- 6.2.1 Initial and Final Permutations**
- 6.2.2 Rounds**
- 6.2.3 Cipher and Reverse Cipher**
- 6.2.4 Examples**

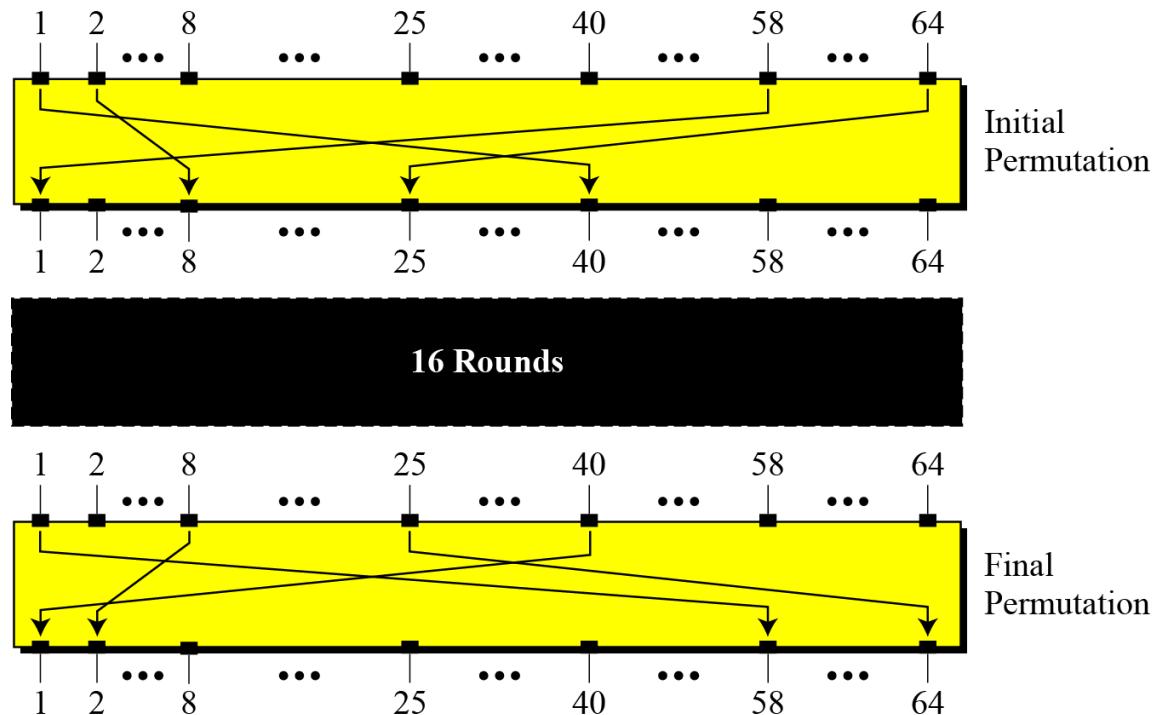
## 6-2 Continue

**Figure 6.2 General structure of DES**



## 6.2.1 Initial and Final Permutations

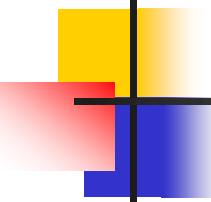
Figure 6.3 Initial and final permutation steps in DES



## 6.2.1 Continue

**Table 6.1** *Initial and final permutation tables*

<i>Initial Permutation</i>	<i>Final Permutation</i>
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25



## *6.2.1 Continued*

### Example 6.1

**Find the output of the final permutation box when the input is given in hexadecimal as:**

0x0000 0080 0000 0002

### Solution

**Only bit 25 and bit 63 are 1s; the other bits are 0s. In the final permutation, bit 25 becomes bit 64 and bit 63 becomes bit 15. The result is**

0x0002 0000 0000 0001

## 6.2.1 *Continued*

### Example 6.2

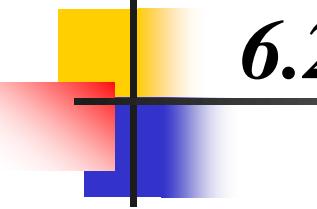
Prove that the initial and final permutations are the inverse of each other by finding the output of the initial permutation if the input is

0x0002 0000 0000 0001

### Solution

The input has only two 1s; the output must also have only two 1s. Using Table 6.1, we can find the output related to these two bits. Bit 15 in the input becomes bit 63 in the output. Bit 64 in the input becomes bit 25 in the output. So the output has only two 1s, bit 25 and bit 63. The result in hexadecimal is

0x0000 0080 0000 0002



## *6.2.1 Continued*

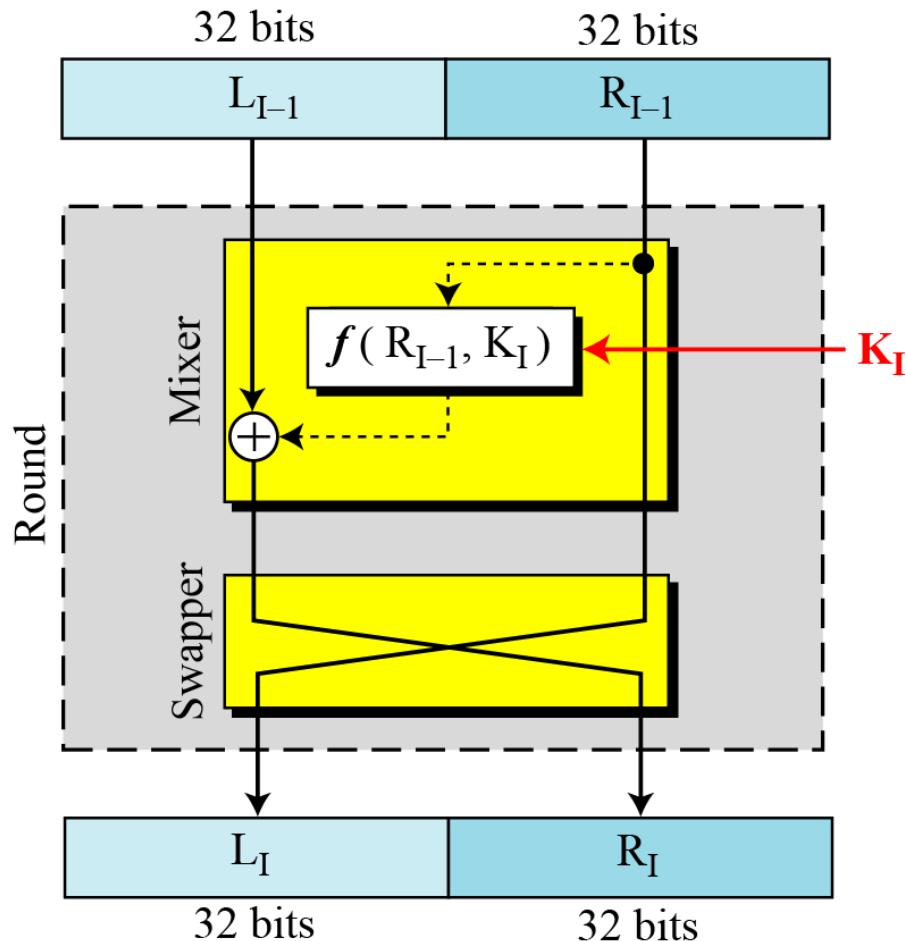
**Note**

**The initial and final permutations are straight P-boxes that are inverses of each other.**

**They have no cryptography significance in DES.**

## 6.2.2 Rounds

*DES uses 16 rounds. Each round of DES is a Feistel cipher.*



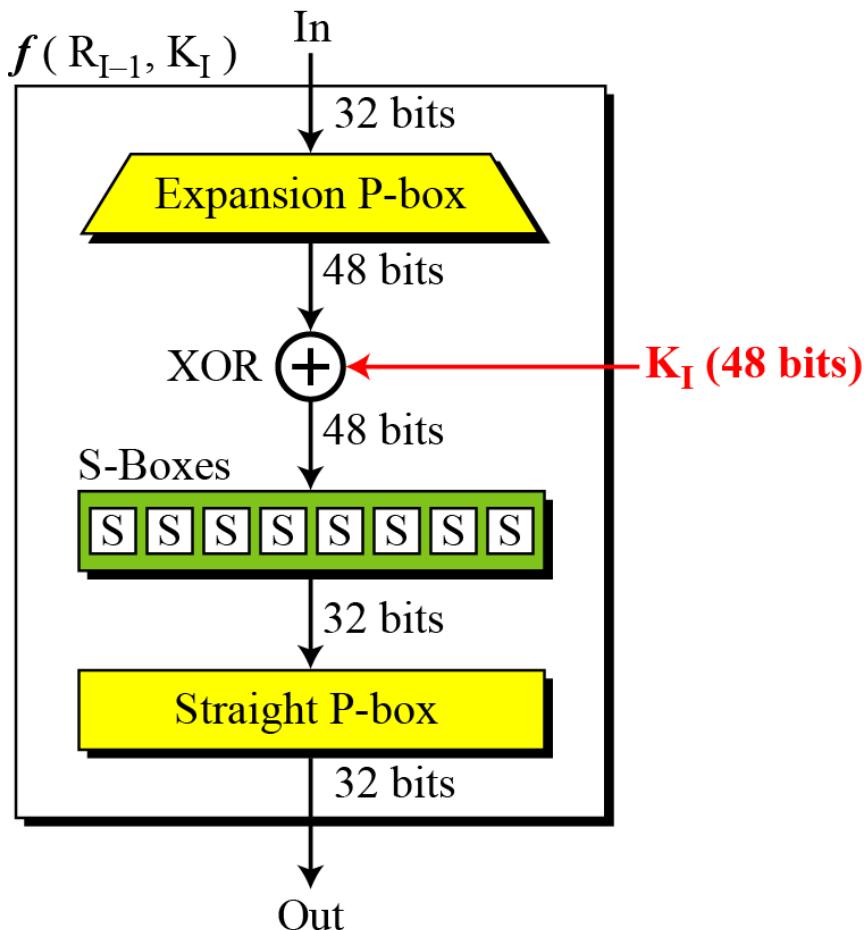
**Figure 6.4**  
A round in DES  
(encryption site)

## 6.2.2 Continued

### DES Function

*The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.*

**Figure 6.5**  
*DES function*

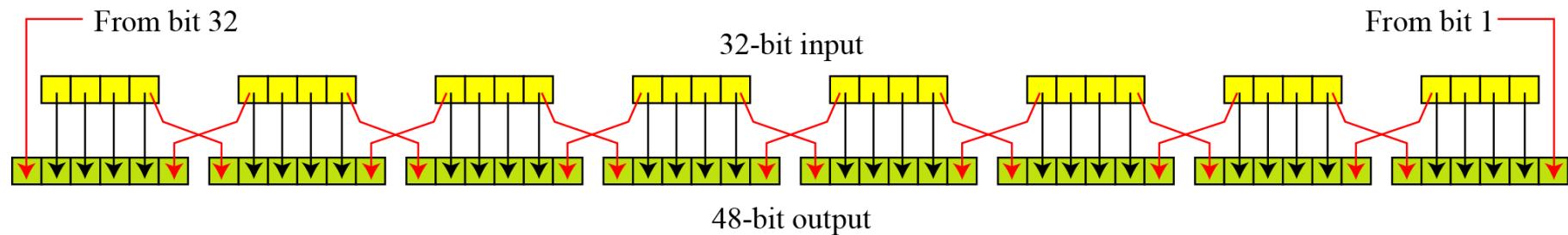


## 6.2.2 Continue

### Expansion P-box

Since  $R_{I-1}$  is a 32-bit input and  $K_I$  is a 48-bit key, we first need to expand  $R_{I-1}$  to 48 bits.

Figure 6.6 Expansion permutation



## 6.2.2 Continue

*Although the relationship between the input and output can be defined mathematically, DES uses Table 6.2 to define this P-box.*

**Table 6.6 Expansion P-box table**

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

## 6.2.2 Continue

### *Whitener (XOR)*

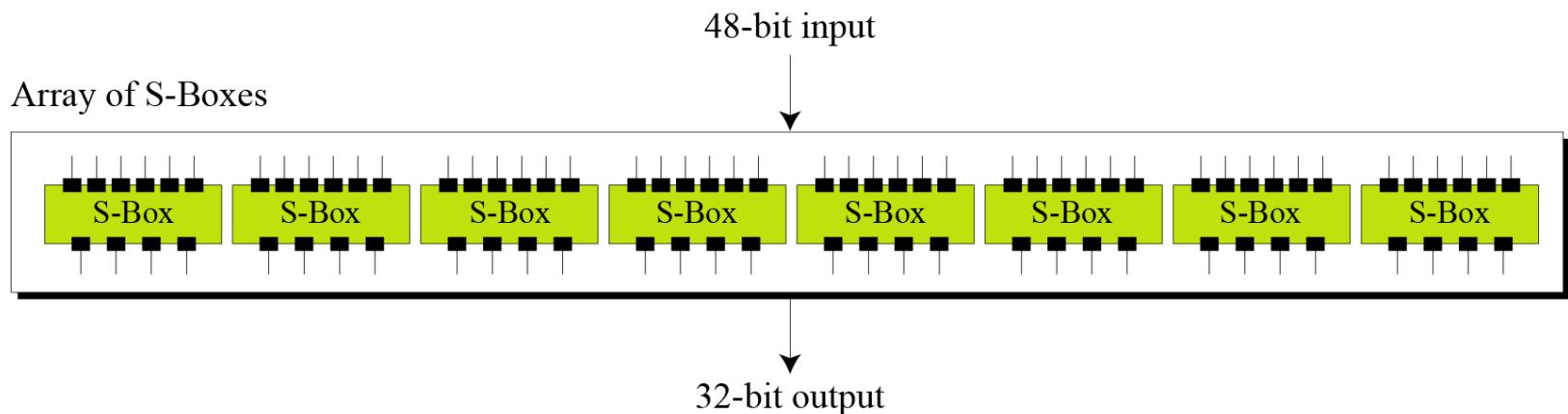
*After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key. Note that both the right section and the key are 48-bits in length. Also note that the round key is used only in this operation.*

## 6.2.2 Continue

### S-Boxes

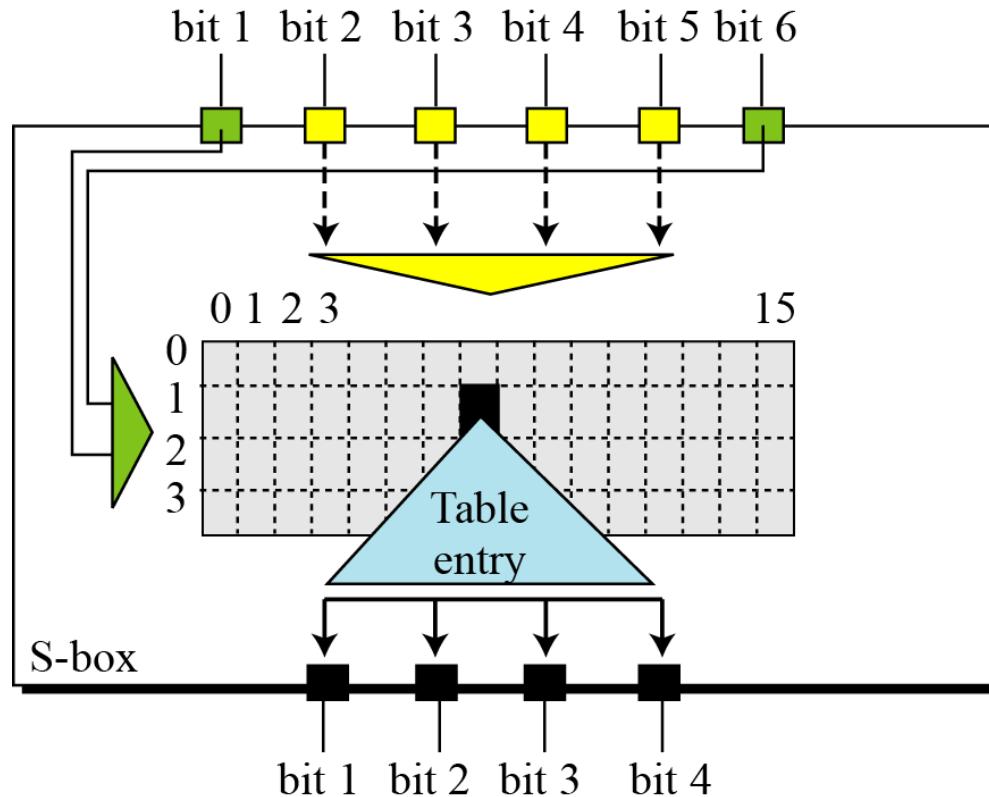
*The S-boxes do the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. See Figure 6.7.*

**Figure 6.7 S-boxes**



## 6.2.2 Continue

Figure 6.8 S-box rule



## 6.2.2 Continue

*Table 6.3 shows the permutation for S-box 1. For the rest of the boxes see the textbook.*

**Table 6.3 S-box 1**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

## 6.2.2 *Continued*

### Example 6.3

The input to S-box 1 is **100011**. What is the output?

#### Solution

If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal. The remaining bits are 0001 in binary, which is 1 in decimal. We look for the value in row 3, column 1, in Table 6.3 (S-box 1). The result is 12 in decimal, which in binary is 1100. So the input **100011** yields the output **1100**.

## 6.2.2 *Continued*

### Example 6.4

The input to S-box 8 is **000000**. What is the output?

#### Solution

If we write the first and the sixth bits together, we get **00** in binary, which is **0** in decimal. The remaining bits are **0000** in binary, which is **0** in decimal. We look for the value in row **0**, column **0**, in Table 6.10 (S-box 8). The result is **13** in decimal, which is **1101** in binary. So the input **000000** yields the output **1101**.

## 6.2.2 Continue

### Straight Permutation

**Table 6.11** *Straight permutation table*

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

## 6.2.3 *Cipher and Reverse Cipher*

*Using mixers and swappers, we can create the cipher and reverse cipher, each having 16 rounds.*

### *First Approach*

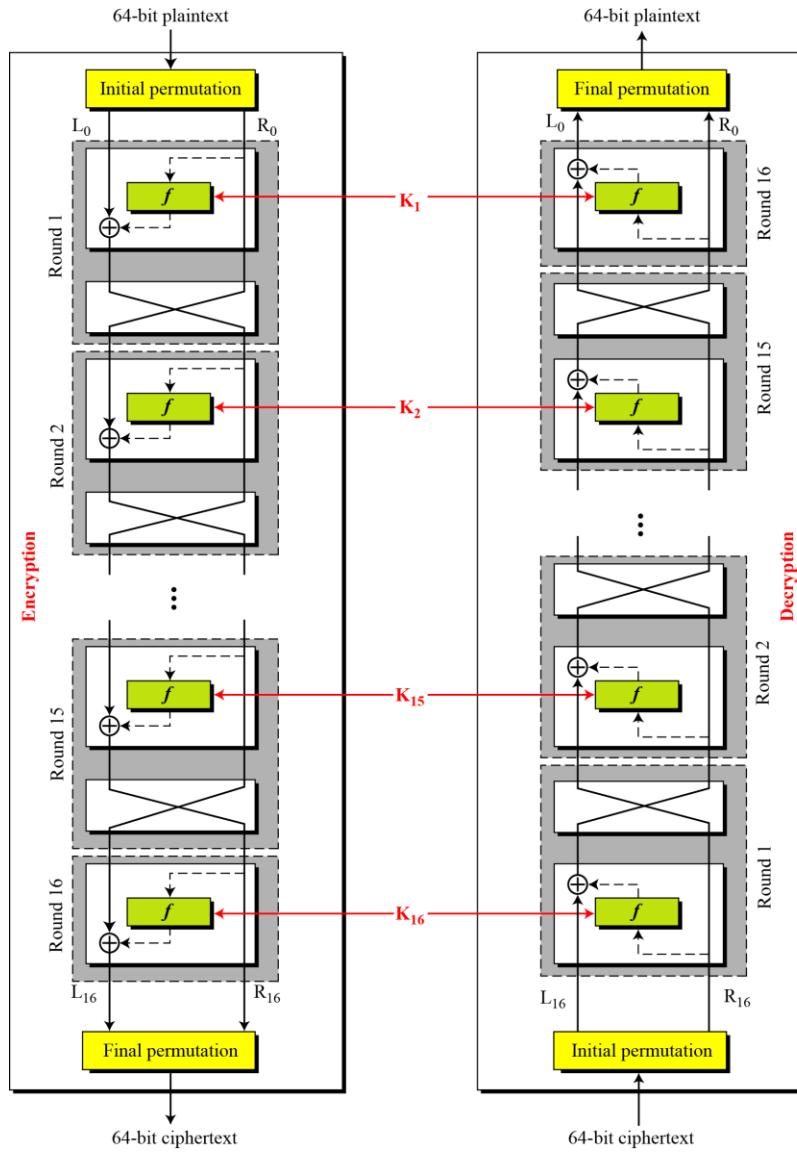
*To achieve this goal, one approach is to make the last round (round 16) different from the others; it has only a mixer and no swapper.*

**Note**

In the first approach, there is no swapper in the last round.

## 6.2.3 Continued

**Figure 6.9 DES cipher and reverse cipher for the first approach**



## 6.2.3 Continued

### Algorithm 6.1 Pseudocode for DES cipher

```
Cipher (plainBlock[64], RoundKeys[16, 48], cipherBlock[64])
{
    permute (64, 64, plainBlock, inBlock, InitialPermutationTable)
    split (64, 32, inBlock, leftBlock, rightBlock)
    for (round = 1 to 16)
    {
        mixer (leftBlock, rightBlock, RoundKeys[round])
        if (round!=16) swapper (leftBlock, rightBlock)
    }
    combine (32, 64, leftBlock, rightBlock, outBlock)
    permute (64, 64, outBlock, cipherBlock, FinalPermutationTable)
}
```

## 6.2.3 *Continued*

### Algorithm 6.1 *Pseudocode for DES cipher (Continued)*

```
mixer (leftBlock[48], rightBlock[48], RoundKey[48])
```

```
{
```

```
    copy (32, rightBlock, T1)
```

```
    function (T1, RoundKey, T2)
```

```
        exclusiveOr (32, leftBlock, T2, T3)
```

```
        copy (32, T3, rightBlock)
```

```
}
```

```
swapper (leftBlock[32], rigthBlock[32])
```

```
{
```

```
    copy (32, leftBlock, T)
```

```
    copy (32, rightBlock, leftBlock)
```

```
    copy (32, T, rightBlock)
```

```
}
```

## 6.2.3 *Continued*

### Algorithm 6.1 *Pseudocode for DES cipher (Continued)*

```
function (inBlock[32], RoundKey[48], outBlock[32])
{
    permute (32, 48, inBlock, T1, ExpansionPermutationTable)
    exclusiveOr (48, T1, RoundKey, T2)
    substitute (T2, T3, SubstituteTables)
    permute (32, 32, T3, outBlock, StraightPermutationTable)
}
```

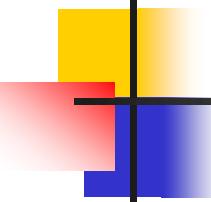
## 6.2.3 Continued

### Algorithm 6.1 Pseudocode for DES cipher (Continued)

```
substitute (inBlock[32], outBlock[48], SubstitutionTables[8, 4, 16])
{
    for (i = 1 to 8)
    {
        row  $\leftarrow$  2  $\times$  inBlock[i  $\times$  6 + 1] + inBlock [i  $\times$  6 + 6]
        col  $\leftarrow$  8  $\times$  inBlock[i  $\times$  6 + 2] + 4  $\times$  inBlock[i  $\times$  6 + 3] +
                  2  $\times$  inBlock[i  $\times$  6 + 4] + inBlock[i  $\times$  6 + 5]

        value = SubstitutionTables [i][row][col]

        outBlock[[i  $\times$  4 + 1]  $\leftarrow$  value / 8;           value  $\leftarrow$  value mod 8
        outBlock[[i  $\times$  4 + 2]  $\leftarrow$  value / 4;           value  $\leftarrow$  value mod 4
        outBlock[[i  $\times$  4 + 3]  $\leftarrow$  value / 2;           value  $\leftarrow$  value mod 2
        outBlock[[i  $\times$  4 + 4]  $\leftarrow$  value
    }
}
```



## *6.2.3 Continued*

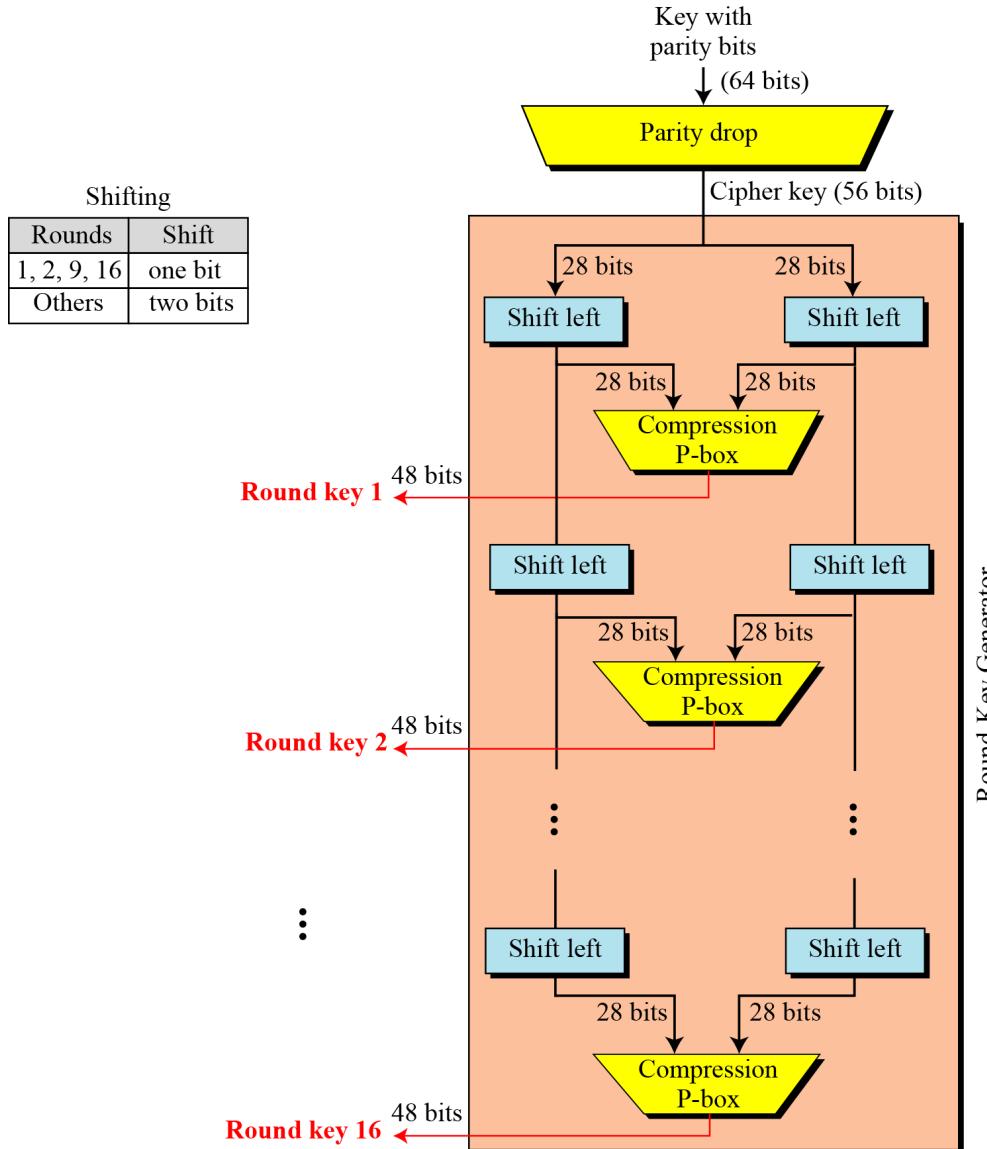
### *Alternative Approach*

*We can make all 16 rounds the same by including one swapper to the 16th round and add an extra swapper after that (two swappers cancel the effect of each other).*

### *Key Generation*

*The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.*

## 6.2.3 Continued



**Figure 6.10**  
**Key generation**

## 6.2.3 *Continued*

**Table 6.12** *Parity-bit drop table*

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

**Table 6.13** *Number of bits shifts*

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

## 6.2.3 *Continued*

**Table 6.14** *Key-compression table*

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

## 6.2.3 *Continued*

### Algorithm 6.2 *Algorithm for round-key generation*

```
Key_Generator (keyWithParities[64], RoundKeys[16, 48], ShiftTable[16])
{
    permute (64, 56, keyWithParities, cipherKey, ParityDropTable)
    split (56, 28, cipherKey, leftKey, rightKey)
    for (round = 1 to 16)
    {
        shiftLeft (leftKey, ShiftTable[round])
        shiftLeft (rightKey, ShiftTable[round])
        combine (28, 56, leftKey, rightKey, preRoundKey)
        permute (56, 48, preRoundKey, RoundKeys[round], KeyCompressionTable)
    }
}
```

## 6.2.3 *Continued*

### Algorithm 6.2 *Algorithm for round-key generation (Continue)*

```
shiftLeft (block[28], numOfShifts)
{
    for (i = 1 to numOfShifts)
    {
        T ← block[1]
        for (j = 2 to 28)
        {
            block [j-1] ← block [j]
        }
        block[28] ← T
    }
}
```

## 6.2.4 Examples

### Example 6.5

We choose a random plaintext block and a random key, and determine what the ciphertext block would be (all in hexadecimal):

Plaintext: 123456ABCD132536

Key: AABB09182736CCDD

CipherText: C0B7A8D05F3A829C

**Table 6.15** Trace of data for Example 6.5

Plaintext: 123456ABCD132536			
After initial permutation: 14A7D67818CA18AD After splitting: $L_0 = 14A7D678$ $R_0 = 18CA18AD$			
Round	Left	Right	Round Key
Round 1	18CA18AD	5A78E394	194CD072DE8C
Round 2	5A78E394	4A1210F6	4568581ABCCE
Round 3	4A1210F6	B8089591	06EDA4ACF5B5
Round 4	B8089591	236779C2	DA2D032B6EE3

## 6.2.4 Continued

### Example 6.5 Continued

**Table 6.15** Trace of data for Example 6.5 (Conintued)

Round 5	236779C2	A15A4B87	69A629FEC913
Round 6	A15A4B87	2E8F9C65	C1948E87475E
Round 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Round 8	A9FC20A3	308BEE97	34F822F0C66D
Round 9	308BEE97	10AF9D37	84BB4473DCCC
Round 10	10AF9D37	6CA6CB20	02765708B5BF
Round 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Round 12	FF3C485F	22A5963B	C2C1E96A4BF3
Round 13	22A5963B	387CCDAA	99C31397C91F
Round 14	387CCDAA	BD2DD2AB	251B8BC717D0
Round 15	BD2DD2AB	CF26B472	3330C5D9A36D
Round 16	19BA9212	CF26B472	181C5D75C66D
After combination: 19BA9212CF26B472			
Ciphertext: C0B7A8D05F3A829C			(after final permutation)

## 6.2.4 Continued

### Example 6.6

Let us see how Bob, at the destination, can decipher the ciphertext received from Alice using the same key. Table 6.16 shows some interesting points.

Ciphertext: C0B7A8D05F3A829C			
After initial permutation: 19BA9212CF26B472			
After splitting: $L_0=19BA9212$ $R_0=CF26B472$			
Round	Left	Right	Round Key
Round 1	CF26B472	BD2DD2AB	181C5D75C66D
Round 2	BD2DD2AB	387CCDAA	3330C5D9A36D
...	...	...	...
Round 15	5A78E394	18CA18AD	4568581ABCCE
Round 16	14A7D678	18CA18AD	194CD072DE8C
After combination: 14A7D67818CA18AD			
Plaintext: 123456ABCD132536		(after final permutation)	

## 6-3 DES ANALYSIS

*Critics have used a strong magnifier to analyze DES.  
Tests have been done to measure the strength of some  
desired properties in a block cipher.*

**Topics discussed in this section:**

**6.3.1 Properties**

**6.3.2 Design Criteria**

**6.3.3 DES Weaknesses**

## 6.3.1 Properties

*Two desired properties of a block cipher are the **avalanche effect** and the **completeness**.*

### Example 6.7

To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.

Plaintext: 0000000000000000

Key: 22234512987ABB23

Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000000

Key: 22234512987ABB23

Ciphertext: 0A4ED5C15A63FEA3

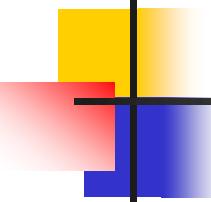
## 6.3.1 *Continued*

### Example 6.7 *Continued*

Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits. This means that changing approximately 1.5 percent of the plaintext creates a change of approximately 45 percent in the ciphertext.

**Table 6.17** *Number of bit differences for Example 6.7*

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit differences	1	6	20	29	30	33	32	29	32	39	33	28	30	31	30	29



## *6.3.1 Continued*

### *Completeness effect*

*Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext.*

## *6.3.2 Design Criteria*

### *S-Boxe*

*The design provides confusion and diffusion of bits from each round to the next.*

### *P-Boxes*

*They provide diffusion of bits.*

### *Number of Rounds*

*DES uses sixteen rounds of Feistel ciphers. the ciphertext is thoroughly a random function of plaintext and ciphertext.*

## 6.3.3 DES Weaknesses

During the last few years critics have found some weaknesses in DES.

### Weaknesses in Cipher Design

1. Weaknesses in S-boxes
2. Weaknesses in P-boxes
3. Weaknesses in Key

**Table 6.18** Weak keys

Keys before parities drop (64 bits)			
0101	0101	0101	0101
1F1F	1F1F	OE0E	OE0E
EOEO	EOEO	F1F1	F1F1
FEFE	FEFE	FEFE	FEFE

Actual key (56 bits)	
0000000	0000000
0000000	FFFFFFF
FFFFFFF	0000000
FFFFFFF	FFFFFFF

### 6.3.3 *Continued*

#### Example 6.8

Let us try the first weak key in Table 6.18 to encrypt a block two times. After two encryptions with the same key the original plaintext block is created. Note that we have used the encryption algorithm two times, not one encryption followed by another decryption.

Key: 0x0101010101010101

Plaintext: 0x1234567887654321

Ciphertext: 0x814FE938589154F7

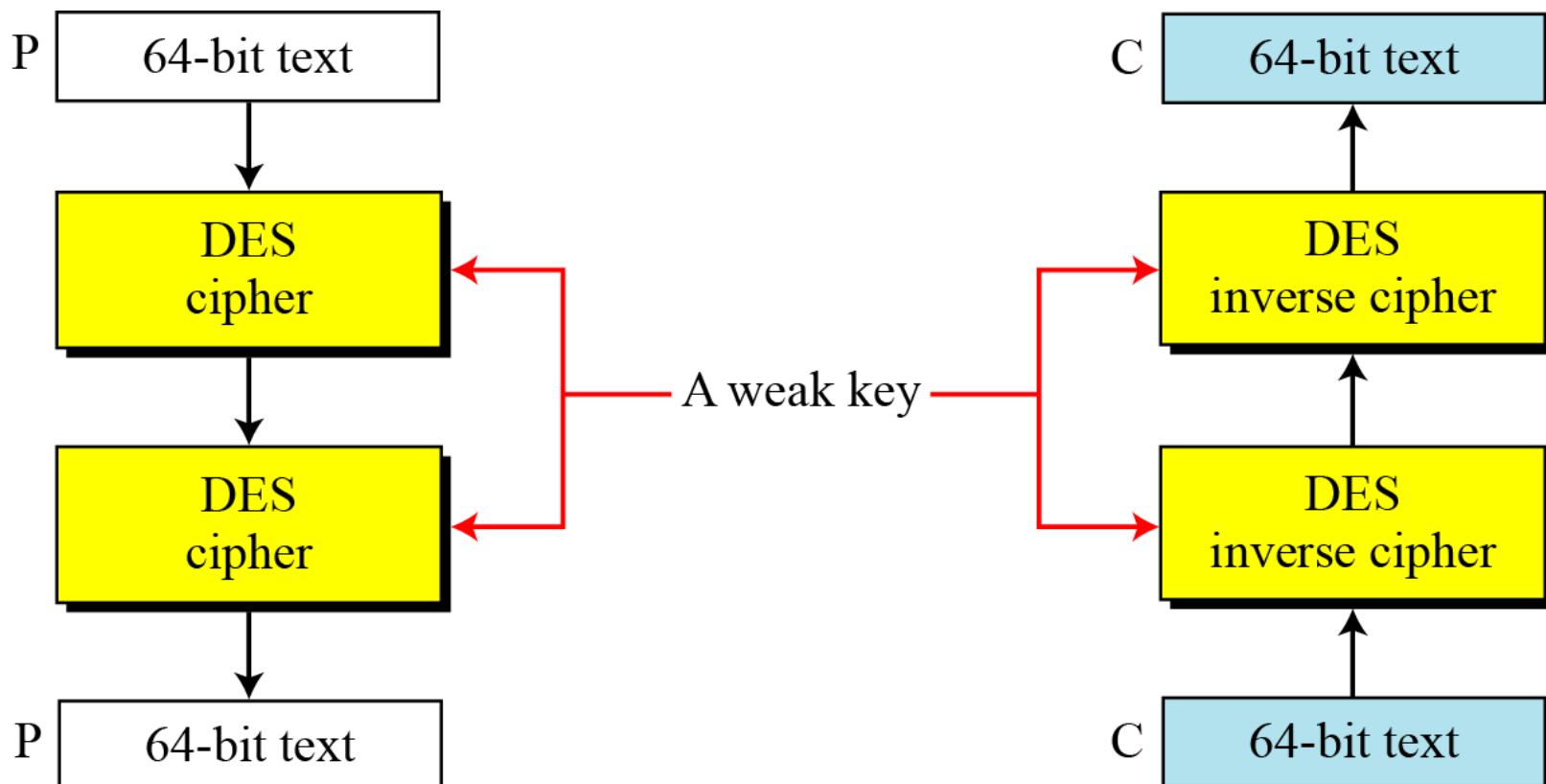
Key: 0x0101010101010101

Plaintext: 0x814FE938589154F7

Ciphertext: 0x1234567887654321

### 6.3.3 Continued

**Figure 6.11 Double encryption and decryption with a weak key**



### 6.3.3 *Continued*

**Table 6.19** *Semi-weak keys*

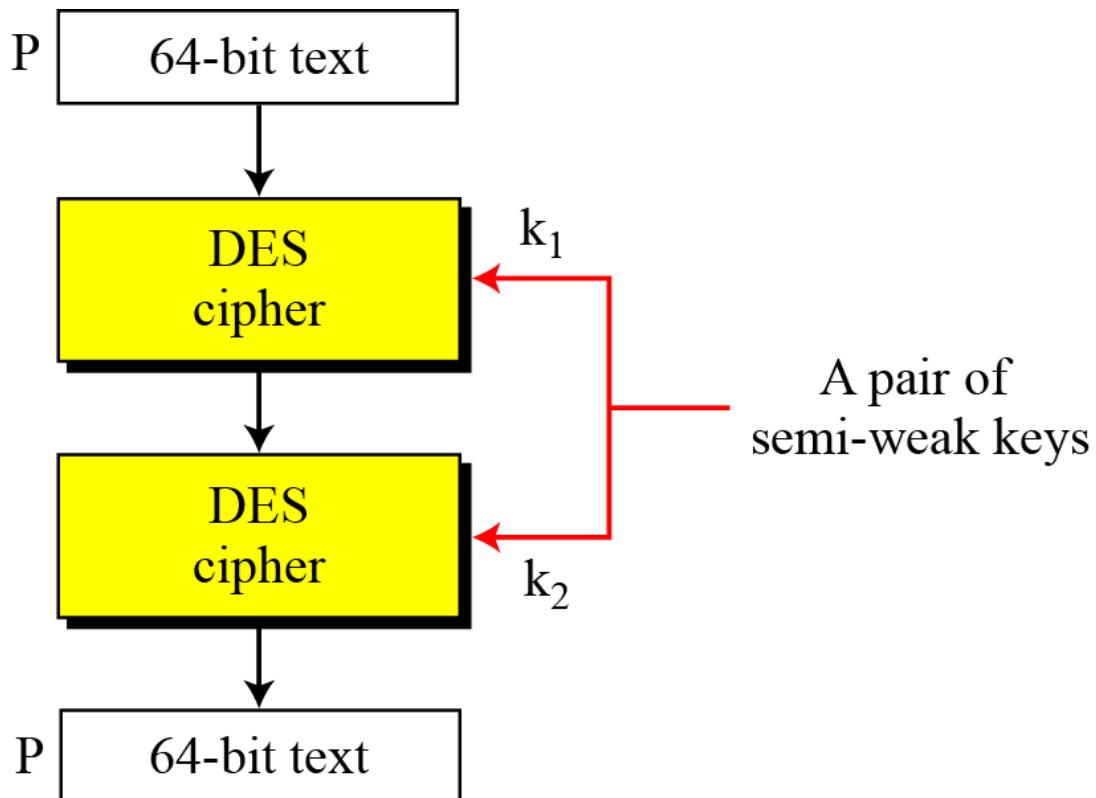
<i>First key in the pair</i>	<i>Second key in the pair</i>
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E1 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
EOF E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

### 6.3.3 *Continued*

<i>Round key 1</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 2</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 3</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 4</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 5</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 6</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 7</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 8</i>	6EAC1ABCE642	9153E54319BD
<i>Round key 9</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 10</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 11</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 12</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 13</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 14</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 15</i>	9153E54319BD	6EAC1ABCE642
<i>Round key 16</i>	6EAC1ABCE642	9153E54319BD

### 6.3.3 Continued

**Figure 6.12** A pair of semi-weak keys in encryption and decryption



### **6.3.3 *Continued***

#### **Example 6.9**

**What is the probability of randomly selecting a weak, a semi-weak, or a possible weak key?**

#### **Solution**

**DES has a key domain of  $2^{56}$ . The total number of the above keys are 64 ( $4 + 12 + 48$ ). The probability of choosing one of these keys is  $8.8 \times 10^{-16}$ , almost impossible.**

### 6.3.3 *Continued*

**Key Complement** In the key domain ( $2^{56}$ ), definitely half of the keys are *complement* of the other half. A **key complement** can be made by inverting (changing 0 to 1 or 1 to 0) each bit in the key. Does a key complement simplify the job of the cryptanalysis? It happens that it does. Eve can use only half of the possible keys ( $2^{55}$ ) to perform brute-force attack. This is because

$$C = E(K, P) \rightarrow \bar{C} = E(\bar{K}, \bar{P})$$

In other words, if we encrypt the complement of plaintext with the complement of the key, we get the complement of the ciphertext. Eve does not have to test all  $2^{56}$  possible keys, she can test only half of them and then complement the result.

### 6.3.3 *Continued*

#### Example 6.10

Let us test the claim about the complement keys. We have used an arbitrary key and plaintext to find the corresponding ciphertext. If we have the key complement and the plaintext, we can obtain the complement of the previous ciphertext (Table 6.20).

**Table 6.20** Results for Example 6.10

	<i>Original</i>	<i>Complement</i>
Key	1234123412341234	EDCBEDCBEDCBEDCB
Plaintext	12345678ABCDEF12	EDCBA987543210ED
Ciphertext	E112BE1DEFC7A367	1EED41E210385C98

## 6-4 Multiple DES

*The major criticism of DES regards its key length. Fortunately DES is not a group. This means that we can use double or triple DES to increase the key size.*

**Topics discussed in this section:**

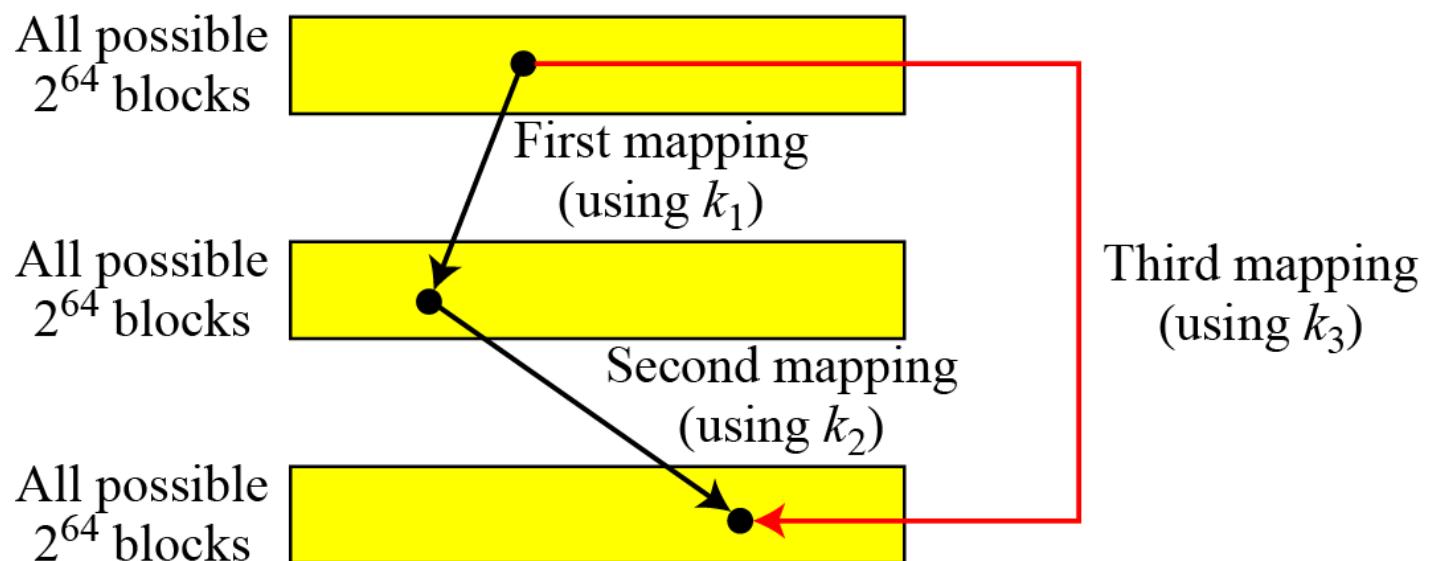
**6.4.1 Double DES**

**6.4.4 Triple DES**

## 6-4 Continued

*A substitution that maps every possible input to every possible output is a group.*

**Figure 6.13** *Composition of mapping*



## **6.4.1 Double DES**

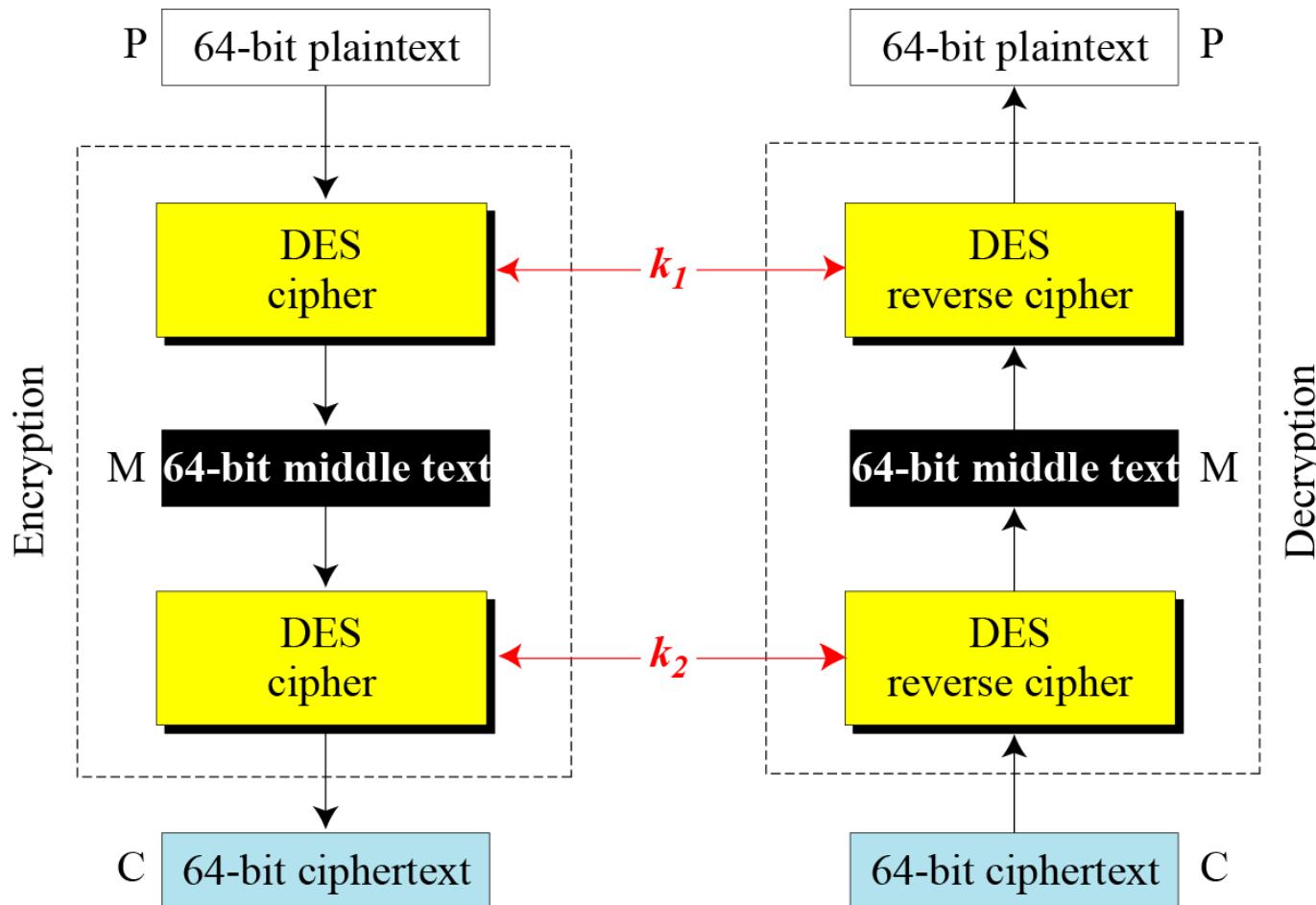
*The first approach is to use double DES (2DES).*

### ***Meet-in-the-Middle Attack***

*However, using a known-plaintext attack called **meet-in-the-middle attack** proves that double DES improves this vulnerability slightly (to  $2^{57}$  tests), but not tremendously (to  $2^{112}$ ).*

## 6.4.1 Continued

Figure 6.14 Meet-in-the-middle attack for double DES



## 6.4.1 Continued

**Figure 6.15** Tables for meet-in-the-middle attack

$$M = E_{k_1}(P)$$

M	$k_1$
●	

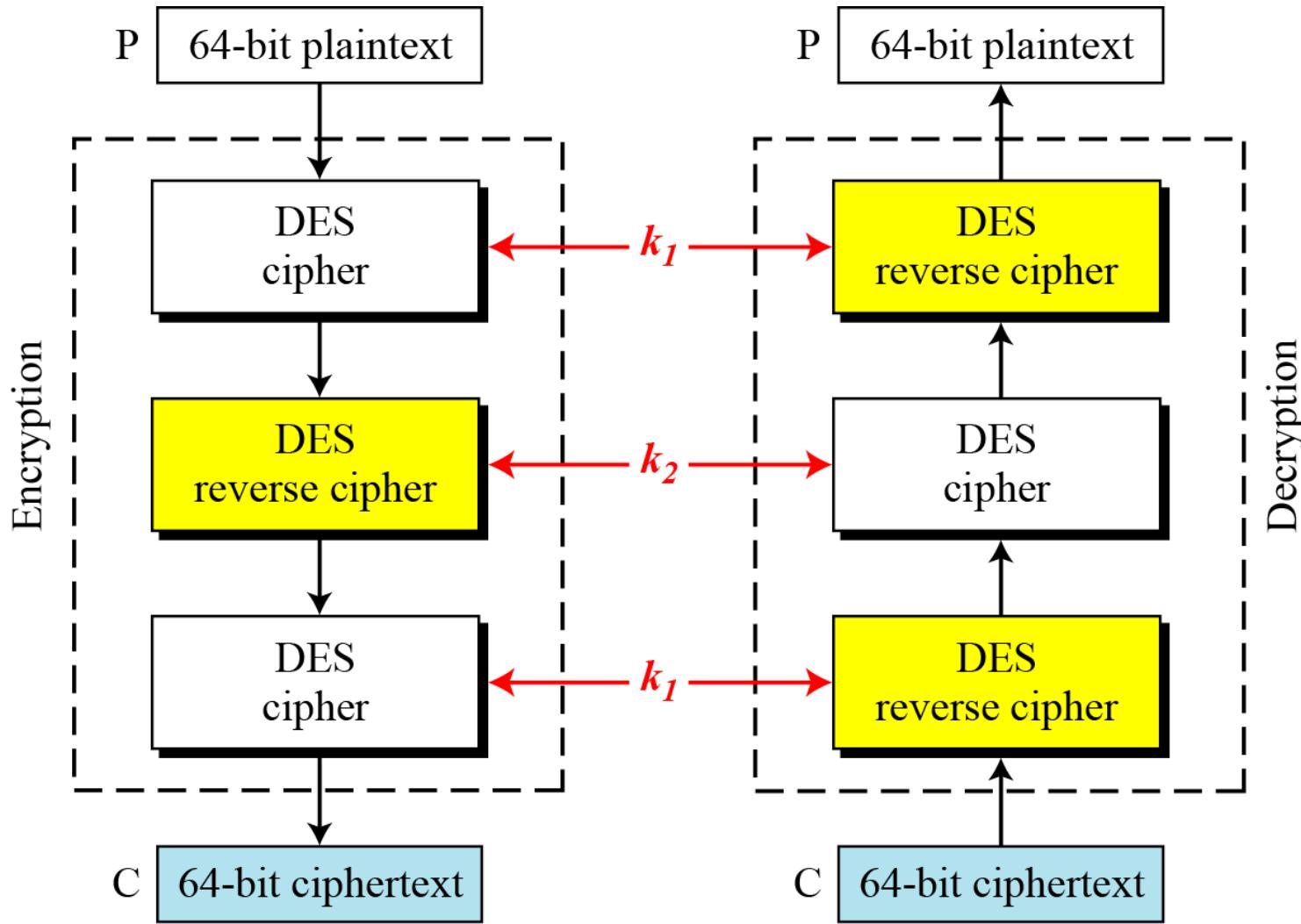
$$M = D_{k_2}(C)$$

M	$k_2$
●	

Find equal M's and record  
corresponding  $k_1$  and  $k_2$

## 6.4.2 *Triple DES*

Figure 6.16 *Triple DES with two keys*



## ***6.4.2 Continuous***

### ***Triple DES with Three Keys***

*The possibility of known-plaintext attacks on triple DES with two keys has enticed some applications to use triple DES with three keys. Triple DES with three keys is used by many applications such as PGP (See Chapter 16).*

## 6-5 Security of DES

*DES, as the first important block cipher, has gone through much scrutiny. Among the attempted attacks, three are of interest: brute-force, differential cryptanalysis, and linear cryptanalysis.*

**Topics discussed in this section:**

**6.5.1 Brute-Force Attack**

**6.5.2 Differential Cryptanalysis**

**6.5.3 Linear Cryptanalysis**

## *6.5.1 Brute-Force Attack*

*We have discussed the weakness of short cipher key in DES. Combining this weakness with the key complement weakness, it is clear that DES can be broken using  $2^{55}$  encryptions.*

## 6.5.2 Differential Cryptanalysis

*It has been revealed that the designers of DES already knew about this type of attack and designed S-boxes and chose 16 as the number of rounds to make DES specifically resistant to this type of attack.*

### Note

We show an example of DES differential cryptanalysis in Appendix N.

## 6.5.3 Linear Cryptanalysis

*Linear cryptanalysis is newer than differential cryptanalysis. DES is more vulnerable to linear cryptanalysis than to differential cryptanalysis. S-boxes are not very resistant to linear cryptanalysis. It has been shown that DES can be broken using  $2^{43}$  pairs of known plaintexts. However, from the practical point of view, finding so many pairs is very unlikely.*

**Note**

We show an example of DES linear cryptanalysis in Appendix N.